```
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMMMMM   MMMMMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRR         RRR  MMM   MMM   MMM  SSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM         MMM     SSSSSSSSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR   RRR        MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR      RRR     MMM         MMM              SSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
RRR         RRR  MMM         MMM  SSSSSSSSSSSS
```

```
_$

Sy
---
NT$
NT$
NT$
NT$
NT$
NT$



NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$
NT$

NT$

NT$
NT$
NT$
NT$
NT$
NT$
NT$



NT$
NT$
NT$
NT$
NT$
NT$
PI
```

```
**FILE**ID**NTOOPEN




NN      NN  TTTTTTTTTT    000000      OOOOOO    PPPPPPPP    EEEEEEEEEE  NN      NN
NN      NN  TTTTTTTTTT    000000      OOOOOO    PPPPPPPP    EEEEEEEEEE  NN      NN
NN      NN      TT      00      00  OO      OO  PP      PP  EE          NN      NN
NN      NN      TT      00      00  OO      OO  PP      PP  EE          NN      NN
NNNN    NN      TT      00    0000  OO      OO  PP      PP  EE          NNNN    NN
NNNN    NN      TT      00    0000  OO      OO  PP      PP  EE          NNNN    NN
NN  NN  NN      TT      00  00  00  OO      OO  PPPPPPPP    EEEEEEEE    NN  NN  NN
NN  NN  NN      TT      00  00  00  OO      OO  PPPPPPPP    EEEEEEEE    NN  NN  NN
NN    NNNN      TT      0000    00  OO      OO  PP          EE          NN    NNNN
NN    NNNN      TT      0000    00  OO      OO  PP          EE          NN    NNNN
NN      NN      TT      00      00  OO      OO  PP          EE          NN      NN      ....
NN      NN      TT      00      00  OO      OO  PP          EE          NN      NN      ....
NN      NN      TT        000000      OOOOOO    PP          EEEEEEEEEE  NN      NN      ....
NN      NN      TT        000000      OOOOOO    PP          EEEEEEEEEE  NN      NN      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

NTC
VO4

Table of contents

```
0000            $BEGIN  NTOOPEN,000,NF$NETWORK,<NETWORK OPEN FILE>
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
```

```
0000    ;++
0000    ; Facility: RMS
0000    ;
0000    ; Abstract:
0000    ;
0000    ;       This module communicates with the File Access Listener (FAL) at the
0000    ;       remote node to open the specified file.
0000    ;
0000    ; Environment: VAX/VMS, executive mode
0000    ;
0000    ; Author: James A. Krycka,      Creation Date:  09-DEC-1977
0000    ;
0000    ; Modified By:
0000    ;
0000    ;       V03-010 JEJ0038         J E Johnson             19-Jun-1984
0000    ;               Check for VAX/VMS or VAXELAN partner before trying to
0000    ;               use the system specific DAP fields.
0000    ;
0000    ;       V03-009 JAK0142         J A Krycka      10-APR-1984
0000    ;               Fix inconsistency in handling MRS value when dealing with a
0000    ;               remote FAL that uses a stream-based file system.
0000    ;
0000    ;       V03-008 JAK0138         J A Krycka      28-MAR-1984
0000    ;               Call modified NT$EXCH_CNF routine with a parameter.
0000    ;               General cleanup.
0000    ;
0000    ;       V03-007 LJK0254         Lawrence J. Kenah       6-Dec-1983
0000    ;               Change LIB$CVT_OTB reference to FIL$CVT_OTB.
0000    ;
0000    ;       V03-006 KRM0091         Karl Malik              18-Mar-1983
0000    ;               Add support for STMLF and STMCR file formats.
0000    ;
0000    ;       V03-005 KBT0411         Keith B. Thompson       30-Nov-1982
0000    ;               Change IFB$W_DEVBUFSIZ to IFB$L_DEVBUFSIZ.
0000    ;
0000    ;       V03-004 JAK0102         J A Krycka      09-OCT-1982
0000    ;               Fix bug in converting DAP OWNER value into binary format.
0000    ;
0000    ;--
```

```
0000          .SBTTL  DECLARATIONS
0000
0000    ;
0000    ; Include Files:
0000    ;
0000
0000          $DAPPLGDEF                            ; Define DAP prologue symbols
0000          $DAPHDRDEF                            ; Define DAP message header
0000          $DAPSSPDEF                            ; Define DAP system specific field
0000          $DAPATTDEF                            ; Define DAP Attributes message
0000          $DAPACCDEF                            ; Define DAP Access message
0000          $DAPCMPDEF                            ; Define DAP Access Complete message
0000          $DAPKEYDEF                            ; Define DAP Key Definition message
0000          $DAPALLDEF                            ; Define DAP Allocation message
0000          $DAPSUMDEF                            ; Define DAP Summary message
0000          $DAPTIMDEF                            ; Define DAP Date and Time message
0000          $DAPPRODEF                            ; Define DAP Protection message
0000          $DAPNAMDEF                            ; Define DAP Name message
0000          $FABDEF                               ; Define File Access Block symbols
0000          $FWADEF                               ; Define File Work Area symbols
0000          $IFBDEF                               ; Define IFAB symbols
0000          $NWADEF                               ; Network Work Area symbols
0000          $XABDEF                               ; Define symbols common to all XABs
0000          $XABALLDEF                            ; Define Allocation XAB symbols
0000          $XABDATDEF                            ; Define Date and Time XAB symbols
0000          $XABFHCDEF                            ; Define File Header Char symbols
0000          $XABKEYDEF                            ; Define Key Definition XAB symbols
0000          $XABPRODEF                            ; Define Protection XAB symbols
0000          $XABRDTDEF                            ; Define Revision Date/Time XAB symbols
0000          $XABSUMDEF                            ; Define Summary XAB symobls
0000
0000    ;
0000    ; Macros:
0000    ;
0000          None
0000    ;
0000    ; Equated Symbols:
0000    ;
0000
0000          ASSUME  DAP$Q_DCODE_FLG EQ 0
0000          ASSUME  NWA$Q_FLG EQ 0
0000
0000    ;
0000    ; Own Storage:
0000    ;
0000          None
0000    ;
```

```
                         0000          .SBTTL  NT$OPEN - PERFORM NETWORK OPEN FUNCTION
                         0000
                         0000  ;++
                         0000  ; NT$OPEN - engages in a DAP dialogue with the remote FAL to open the
                         0000  ;         specified sequential, relative, or indexed file.
                         0000  ;
                         0000  ; Calling Sequence:
                         0000  ;
                         0000  ;         BSBW    NT$OPEN
                         0000  ;
                         0000  ; Input Parameters:
                         0000  ;
                         0000  ;         R8      FAB address
                         0000  ;         R9      IFAB address
                         0000  ;         R10     FWA address
                         0000  ;         R11     Impure Area address
                         0000  ;
                         0000  ; Implicit Inputs:
                         0000  ;
                         0000  ;         User FAB
                         0000  ;         DAP$V_FCS
                         0000  ;         DAP$V_STM_ONLY
                         0000  ;         DAP$V_VAXVMS
                         0000  ;         IFB fields
                         0000  ;
                         0000  ; Output Parameters:
                         0000  ;
                         0000  ;         R0      Status code (RMS)
                         0000  ;         R1-R7   Destroyed
                         0000  ;         AP      Destroyed
                         0000  ;
                         0000  ; Implicit Outputs:
                         0000  ;
                         0000  ;         User FAB
                         0000  ;         User ALL, DAT, FHC, KEY, PRO, RDT, and SUM XABs
                         0000  ;         IFB$V_DAP_OPEN
                         0000  ;         IFB fields
                         0000  ;         NWA$B_RFM
                         0000  ;
                         0000  ; Completion Codes:
                         0000  ;
                         0000  ;         Standard RMS completion codes
                         0000  ;
                         0000  ; Side Effects:
                         0000  ;
                         0000  ;         None
                         0000  ;
                         0000  ;--
                         0000
                         0000  NT$OPEN::                               ; Entry point
                         0000          $TSTPT  NTOPEN                  ;
     57   3C A9   D0     0006          MOVL    IFB$L_NWA_PTR(R9),R7    ; Get address of NWA (and DAP)
                         000A
                         000A  ;+
                         000A  ; Perform the following FOP field processing:
                         000A  ;   (1) disallow the FOP option NFS.
                         000A  ;   (2) disallow the FOP options KFO and UFO set individually, but allow
```

```
                                000A      ;       <KFO!UFO> to support the '$ RUN node::file.exe' DCL command.
                                000A      ;   (3) ignore the FOP option DFW without returning an error as DFW is an
                                000A      ;       unsupported performance option.
                                000A      ;-
                                000A
          50    04 A8     D0    000A              MOVL    FAB$L_FOP(R8),R0          ; Get user FOP options
          15 50    10     E0    000E              BBS     #FAB$V_NFS,R0,20$         ; Declare this option unsupported
          08 50    1E     E1    0012              BBC     #FAB$V_KFO,R0,10$         ; These options are each unsupported
          04 50    11     E1    0016              BBC     #FAB$V_UFO,R0,10$         ;  unless both are set
                   50     D4    001A              CLRL    R0                        ; Denote non-standard type of access
                   0F     11    001C              BRB     40$                       ; Consider this a load image function
   50    40020000 8F      D3    001E      10$:    BITL    #<<FAB$M_KFO>!-           ; Branch if none are specified;
                                0025                        <FAB$M_UFO>!-           ;  otherwise declare these options
                                0025                      0>,R0                     ;  unsupported over the network
                03 13             0025              BEQL    30$                       ;
             FFD6'  31           0027      20$:    BRW     NT$LCL_FOP                ; Return RMS$_SUPPORT error and
                                002A                                                ;  exit with RMS code in R0
                                002A
                                002A      ;+
                                002A      ; Exchange DAP Configuration messages with FAL and determine DAP buffer size.
                                002A      ;-
                                002A
          50    01        D0    002A      30$:    MOVL    #DAP$K_OPEN,R0            ; Denote type of file access
             FFD0'        30    002D      40$:    BSBW    NT$EXCH_CNF               ; Exchange Configuration messages
             01 50        E8    0030              BLBS    R0,BUILD_MASK             ; Branch on success
                   05           0033      FAIL1:  RSB                               ; Exit with RMS code in R0
                                0034
                                0034      ;+
                                0034      ; Build display field request mask which will be used in the Access message
                                0034      ; to request that optional DAP messages be returned by FAL. For $OPEN, these
                                0034      ; are the ALL, KEY, PRO, SUM, TIM, and NAM messages. (Note that the Attributes
                                0034      ; message is required which will supply information to update both the FAB and
                                0034      ; FHCXAB.)
                                0034      ;-
                                0034
                                0034      BUILD_MASK:                               ; Build NWA$W_DISPLAY
                   56     D4    0034              CLRL    R6                        ; Indicate this is not a close operation
             FFC7'        30    0036              BSBW    NT$SCAN_XABCHN            ; Scan user XAB chain and check FAL's
                                0039                                                ;  capabilities; request mask put in R2
             F7 50        E9    0039              BLBC    R0,FAIL1                  ; Branch on failure to complete scan
             FFC1'        30    003C              BSBW    NT$SCAN_NAMBLK            ; Scan user NAM block and check FAL's
                                003F                                                ;  capabilities; update request mask
             F1 50        E9    003F              BLBC    R0,FAIL1                  ; Branch on failure to complete scan
          52    01        A8    0042              BISW2   #DAP$M_DSP_ATT,R2         ; Request main Attributes message
     00D0 C7    52        B0    0045              MOVW    R2,NWA$W_DISPLAY(R7)      ; Save request mask
                                004A
                                004A      ;+
                                004A      ; Build and send DAP Attributes message to partner.
                                004A      ;-
                                004A
                                004A      SEND_ATT:                                 ; (required message)
          50    02        D0    004A              MOVL    #DAP$K_ATT_MSG,R0         ; Get message type value
             FFB0'        30    004D              BSBW    NT$BUILD_HEAD             ; Construct message header
                                0050
                                0050              ASSUME  DAP$K_SEQ EQ FAB$C_SEQ
                                0050              ASSUME  DAP$K_REL EQ FAB$C_REL
                                0050              ASSUME  DAP$K_IDX EQ FAB$C_IDX
```

```
                        0050
                        0050          ASSUME  DAP$K_UDF EQ FAB$C_UDF
                        0050          ASSUME  DAP$K_FIX EQ FAB$C_FIX
                        0050          ASSUME  DAP$K_VAR EQ FAB$C_VAR
                        0050          ASSUME  DAP$K_VFC EQ FAB$C_VFC
                        0050          ASSUME  DAP$K_STM EQ FAB$C_STM
                        0050          ASSUME  DAP$K_STMLF EQ FAB$C_STMLF
                        0050          ASSUME  DAP$K_STMCR EQ FAB$C_STMCR
                        0050
                        0050          ASSUME  DAP$V_FTN EQ FAB$V_FTN
                        0050          ASSUME  DAP$V_CR  EQ FAB$V_CR
                        0050          ASSUME  DAP$V_PRN EQ FAB$V_PRN
                        0050          ASSUME  DAP$V_BLK EQ FAB$V_BLK
                        0050
                        0050          ASSUME  DAP$K_DATATYP_D EQ DAP$M_IMAGE
                        0050          ASSUME  DAP$K_ORG_D EQ DAP$K_SEQ
                        0050          ASSUME  DAP$K_RFM_D EQ DAP$K_FIX
                        0050          ASSUME  DAP$K_RAT_D EQ DAP$M_EMBEDDED
                        0050
                        0050  ;
                        0050  ; Construct attriburtes menu mask.
                        0050  ;
                        0050
51    1020 8F      3C   0050  PART1:  MOVZWL  #<<DAP$M_MRS>!-              ; Always include MRS and FOP fields
                        0055                    <DAP$M_FOP1>!-             ;  in mask
                        0055                  0>,R1                        ;
           05      E0   0055          BBS     #FAB$V_BIO,-                 ; Branch if block I/O mode
      11 16 A8          0057                  FAB$B_FAC(R8),10$            ;
      51   0F      C8   005A          BISL2   #<<DAP$M_DATATYPE>!-         ; Add DATATYPE, ORG, RFM, and RAT fields
                        005D                    <DAP$M_ORG>!-              ;  to mask
                        005D                    <DAP$M_RFM>!-              ;
                        005D                    <DAP$M_RAT>!-              ;
                        005D                  0>,R1                        ;
   0A 67   34      E1   005D          BBC     #DAP$V_VAXVMS,(R7),10$       ; Branch if partner is not VAX/VMS
   1F A8   03      91   0061          CMPB    #FAB$C_VFC,FAB$B_RFM(R8);    ; Check for VFC format
           04      12   0065          BNEQ    10$                          ;
                        0067          $SETBIT #DAP$V_FSZ,R1                ; Add FSZ field to mask
      14 A8        B5   006B  10$:    TSTW    FAB$W_DEQ(R8)                ; Branch if DEQ = 0
           04      13   006E          BEQL    20$                          ;
                        0070          $SETBIT #DAP$V_DEQ1,R1               ; Add DEQ field to mask
   56      51      D0   0074  20$:    MOVL    R1,R6                        ; Save attributes menu field
         FF86'     30   0077          BSBW    NT$CVT_BN4_EXT               ; Store ATTMENU as an extensible field
                        007A
                        007A  ;
                        007A  ; Store rest of fields per attributes menu.
                        007A  ;
                        007A  ; Note the following DAP field defaults:
                        007A  ;       DATATYPE = IMAGE
                        007A  ;       ORG      = SEQ
                        007A  ;       RFM      = FIX
                        007A  ;       RAT      = EMBEDDED
                        007A  ;
                        007A
   67      32      E1   007A          BBC     #DAP$V_STM_ONLY,(R7),-       ; Branch if not stream-based remote
           1F           007D                  PART2B                       ;  file system
                        007E
                        007E  ;
```

```
                    007E  ; The remote node is using a stream-based file system.
                    007E  ;
                    007E  ; This section deals with the DATATYPE, ORG, RFM, RAT, and MRS fields.
                    007E  ;
                    007E
          05    E0  007E  PART2A: BBS     #FAB$V_BIO,-                  ; Branch if block I/O mode
       13 16 A8     0080                  FAB$B_FAC(R8),10$              ;
       85  01   90  0083          MOVB    #DAP$M_ASCII,(R5)+            ; Store DATATYPE field
       85  00   90  0086          MOVB    #DAP$K_SEQ,(R5)+              ; Store ORG field
       85  04   90  0089          MOVB    #DAP$K_STM,(R5)+              ; Store RFM field
       85  10   90  008C          MOVB    #DAP$M_EMBEDDED,(R5)+         ; Store RAT field
 85   0202 8F   B0  008F          MOVW    #<512+2>,(R5)+                ; Store MRS field (allow for CRLF which
                    0094                                                ;  partner may consider part of record)
          35    11  0094          BRB     PART3                         ; Join common code
 85   0200 8F   B0  0096  10$:    MOVW    #512,(R5)+                    ; Store MRS field
          2E    11  009B          BRB     PART3                         ; Join common code
                    009D
                    009D  ;
                    009D  ; The remote node is NOT using a stream-based file system.
                    009D  ;
                    009D  ; This section deals with the DATATYPE, ORG, RFM, RAT, and MRS fields.
                    009D  ;
                    009D
          05    E0  009D  PART2B: BBS     #FAB$V_BIO,-                  ; Branch if block I/O mode
       1E 16 A8     009F                  FAB$B_FAC(R8),20$              ;
       85  01   90  00A2          MOVB    #DAP$M_ASCII,(R5)+            ; Store DATATYPE field
    0B 67  34   E0  00A5          BBS     #DAP$V_VAXVMS,(R7),10$        ; Branch if partner is VAX/VMS
       85  00   90  00A9          MOVB    #DAP$K_SEQ,(R5)+              ; Store ORG field
       85  02   90  00AC          MOVB    #DAP$K_VAR,(R5)+              ; Store RFM field
       85  02   90  00AF          MOVB    #DAP$M_CR,(R5)+               ; Store RAT field
           0C   11  00B2          BRB     20$                           ;
 85     1D A8   90  00B4  10$:    MOVB    FAB$B_ORG(R8),(R5)+           ; Store ORG field
 85     1F A8   90  00B8          MOVB    FAB$B_RFM(R8),(R5)+           ; Store RFM field
 85     1E A8   90  00BC          MOVB    FAB$B_RAT(R8),(R5)+           ; Store RAT field
           85   B4  00C0  20$:    CLRW    (R5)+                         ; Zero MRS field
    05 67  31   E1  00C2          BBC     #DAP$V_FCS,(R7),PART3         ; Branch if partner is not FCS based
FE A5   48 A9   B0  00C6          MOVW    IFB$L_DEVBUFSIZ(R9),-2(R5); Specify maximum value
                    00CB
                    00CB  ;
                    00CB  ; This section deals with the FSZ, DEQ, and FOP fields.
                    00CB  ;
                    00CB
  0A 56   08    E1  00CB  PART3:  BBC     #DAP$V_FSZ,R6,10$             ; Used only if RFM = VFC
  85    3F A8   90  00CF          MOVB    FAB$B_FSZ(R8),(R5)+           ; Store FSZ field
          04    12  00D3          BNEQ    10$                           ; Branch if non-zero
  FF A5   02    90  00D5          MOVB    #2,-1(R5)                     ; Use default FSZ value
  04 56   0B    E1  00D9  10$:    BBC     #DAP$V_DEQ1,R6,20$            ; Used only if DEQ > 0
  85    14 A8   B0  00DD          MOVW    FAB$W_DEQ(R8),(R5)+           ; Store DEQ field
        FF1C'   30  00E1  20$:    BSBW    NT$MAP_FOP                    ; Store FOP field
        FF19'   30  00E4          BSBW    NT$BUILD_TAIL                 ; Finish building message
        FF16'   30  00E7          BSBW    NT$TRANSMIT                   ; Send Attributes message to FAL
     50 50      E9  00EA          BLBC    R0,FAIL2                      ; Branch on failure
```

```
                        00ED   ;+
                        00ED   ; Build and send DAP Access message to partner.
                        00ED   ;-
                        00ED
                        00ED   SEND_ACC:                                    ; (required message)
                        00ED            $SETBIT #NWA$V_LAST_MSG,(R7)       ; Declare this last message to block
       50    03    D0   00F1            MOVL    #DAP$K_ACC_MSG,R0          ; Get message type value
          FF09'    30   00F4            BSBW    NT$BUILD_HEAD              ; Construct message header
             11    E1   00F7            BBC     #FAB$V_UFO,-               ; Branch if this is not a load image
     15 04   A8         00F9                    FAB$L_FOP(R8),10$          ;  function (for image activator)
     04 67   34    E0   00FC            BBS     #DAP$V_VAXVMS,(R7),5$      ; It is a VMS partner so we can go on
        67   35    E1   0100            BBC     #DAP$V_VAXELAN,(R7),-
             3A         0103                    FAIL_FOP                   ; It is not ELAN partner so we must fail
     FF A5   20    88   0104   5$:      BISB2   #DAP$M_SYSPEC,-1(R5)       ; Modify flags field
        85   02    90   0108            MOVB    #2,(R5)+                   ; Store SYSPEC as an image field
        85   02    90   010B            MOVB    #DAP$M_SSP_FLG,(R5)+       ; Store SSP_MENU sub-field
        85   01    90   010E            MOVB    #DAP$M_LOAD,(R5)+          ; Store SSP_FLG sub-field
                        0111                                               ; Message header is now complete ...
        85   01    90   0111   10$:     MOVB    #DAP$K_OPEN,(R5)+          ; Store ACCFUNC field
        85   01    90   0114            MOVB    #DAP$M_NONFATAL,(R5)+      ; Store ACCOPT field
          FEE6'    30   0117            BSBW    NT$CRC_INIT                ; Initialize CRC value if both parties
                        011A                                               ;  support file level CRC computation
        04   50    E9   011A            BLBC    R0,20$                     ; Branch if CRC checking disabled
     FF A5   08    88   011D            BISB2   #DAP$M_RET_CRC,-1(R5)      ; Request CRC checksum option
          FEDC'    30   0121   20$:     BSBW    NT$GET_FILESPEC            ; Store FILESPEC as a counted
                        0124                                               ;  ASCII string
          FED9'    30   0124            BSBW    NT$GET_FAC_SHR             ; Store FAC and SHR fields
 51    00D0 C7     3C   0127            MOVZWL  NWA$W_DISPLAY(R7),R1       ; Get request mask
        51   01    B1   012C            CMPW    #DAP$M_DSP_ATT,R1          ; Omit DISPLAY field from message if
             03    13   012F            BEQL    30$                        ;  only Attributes message specified
                        0131                                               ;  (because some older FALs do not
                        0131                                               ;  support this field nor Ext Att msgs)
          FECC'    30   0131            BSBW    NT$CVT_BN4_EXT             ; Store DISPLAY as an extensible field
          FEC9'    30   0134   30$:     BSBW    NT$BUILD_TAIL              ; Finish building message
          FEC6'    30   0137            BSBW    NT$TRANSMIT                ; Send Access message to FAL
        04   50    E8   013A            BLBS    R0,RECV_ATT                ; Branch on success
                   05   013D   FAIL2:   RSB                                ; Exit with RMS code in R0
                        013E   FAIL_FOP:
          FEBF'    31   013E            BRW     NT$LCL_FOP
```

```
                    0141  ;+
                    0141  ; Receive DAP Attributes message from partner and update the user FAB and
                    0141  ; FHCXAB. Also update the user ALLXAB if an Allocation message will not be
                    0141  ; returned by FAL.
                    0141  ;
                    0141  ; Note: The user XAB chain is scanned again to probe all user XABs to protect
                    0141  ;       RMS from a user who deletes the address space where an XAB was
                    0141  ;       previously found.
                    0141  ;-
                    0141
                    0141  RECV_ATT:                                   ; (required message)
                    0141          $SETBIT #DAP$K_ATT_MSG,DAP$L_MSG_MASK(R7)
                    0146                                              ; Expect response of Attributes message
      FEB7'   30    0146          BSBW    NT$RECEIVE                  ; Get reply from FAL
      62 50   E9    0149          BLBC    R0,FAIL3                    ; Branch on failure
      45 A7   90    014C          MOVB    DAP$B_ORG(R7),-             ; Save file organization type
    00C6 C7         014F                  NWA$B_ORG(R7)               ;
      46 A7   90    0152          MOVB    DAP$B_RFM(R7),-             ; Save file format type
    00C7 C7         0155                  NWA$B_RFM(R7)               ;
   08 67   32   E1  0158          BBC     #DAP$V_STM_ONLY,(R7),10$; Return an MRS value of zero if partner
           05   E0  015C          BBS     #FAB$V_BIO,-                ;  uses a stream-based file system and
   03 16 A8         015E                  FAB$B_FAC(R8),10$           ;  this is not a block I/O operation
      4A A7   B4    0161          CLRW    DAP$W_MRS(R7)               ;  because remote parrots what we sent
      FE99'   30    0164  10$:    BSBW    NT$MAP_DEV_CHAR             ; Process device characteristics
                    0167
                    0167  ;
                    0167  ; Update user control blocks.
                    0167  ;
                    0167
         56   D4    0167          CLRL    R6                          ; Indicate this is not a close operation
      FE94'   30    0169          BSBW    NT$SCAN_XABCHN              ; Scan user XAB list again
      3F 50   E9    016C          BLBC    R0,FAIL3                    ; Branch on failure to scan XABs
       0117   30    016F          BSBW    OPEN_UPDATE_FAB             ; Update user FAB
       01A0   30    0172          BSBW    NT$UPDATE_FHC               ; Update user FHCXAB
         02   E0    0175          BBS     #DAP$V_DSP_ALL,-            ; Branch if Allocation message
    00D0 C7         0177                  NWA$W_DISPLAY(R7),-         ;  was requested
         03         017A                  RECV_EXT_ATT
       0360   30    017B          BSBW    NT$DECODE_ALL_A             ; Update user ALLXAB from fields in
                    017E                                              ;  Attributes message (unless ORG = IDX)
                    017E
                    017E  ;+
                    017E  ; Receive DAP Extended Attributes messages from partner and update the user
                    017E  ; ALL, DAT, KEY, PRO, RDT, and SUM XABs.
                    017E  ;-
                    017E
                    017E  RECV_EXT_ATT:                               ; (optional--must be requested)
         71   10    017E          BSBB    NT$RECV_EXT_ATT             ; Process Extended Attributes messages
      2B 50   E9    0180          BLBC    R0,FAIL3                    ; Branch on failure
                    0183
                    0183  ;+
                    0183  ; Receive DAP (resultant) Name message from partner.
                    0183  ;-
                    0183
                    0183  RECV_NAM:                                   ; (optional--must be requested)
         08   E1    0183          BBC     #DAP$V_DSP_NAM,-            ; Branch if Name message was not
    00D0 C7         0185                  NWA$W_DISPLAY(R7),-         ;  requested
         0E         0188                  RECV_ACK                    ;
```

```
                      0189          $SETBIT #DAP$K_NAM_MSG,DAP$L_MSG_MASK(R7)
                      018E                                         ; Expect response of Name message
      FE6F'   30      018E          BSBW    NT$RECEIVE             ; Get reply from FAL
      1A 50   E9      0191          BLBC    R0,FAIL3               ; Branch on failure
       0469   30      0194          BSBW    NT$DECODE_NAM          ; Process resultant name string
                      0197
                      0197  ;+
                      0197  ; Receive DAP Acknowledge message from partner.
                      0197  ;-
                      0197
                      0197  RECV_ACK:                              ; (required message)
                      0197          $SETBIT #DAP$K_ACK_MSG,DAP$L_MSG_MASK(R7)
                      019C                                         ; Expect response of Acknowledge message
      FE61'   30      019C          BSBW    NT$RECEIVE             ; Get reply from FAL
      0C 50   E9      019F          BLBC    R0,FAIL3               ; Branch on failure
         06   E0      01A2          BBS     #IFB$V_BRO,-           ; Branch if BRO option set
   08 22 A9           01A4                  IFB$B_FAC(R9),FAC_BRO  ;
                      01A7  SUC:    $SETBIT #IFB$V_DAP_OPEN,(R9)   ; Denote FAL has opened file
                      01AB          RMSSUC                         ; Return success
              05      01AE  FAIL3:  RSB                            ; Exit with RMS code in R0
```

```
                          01AF          .SBTTL  FAC_BRO
                          01AF
                          01AF    ;++
                          01AF    ; Convert FAB$V_BRO request to FAB$V_BIO request if partner node is not VMS
                          01AF    ; and the file opened is a relative or indexed file. This is done to facilitate
                          01AF    ; the transfer of such files from a non-VMS system in block I/O mode. See
                          01AF    ; comments for the NT$RET_DEV_CHAR routine for related information.
                          01AF    ;
                          01AF    ; Note: The FAB$V_BRO and RAB$V_BIO options are fully supported VMS to VMS and
                          01AF    ;       and documented as such. However, these options are documented as being
                          01AF    ;       unsupported when communicating with a non-VMS partner. Use of these
                          01AF    ;       options with a non-VMS partner is strictly for Digital component use
                          01AF    ;       only as their behavior in a heterogenoes environment may change in the
                          01AF    ;       future.
                          01AF    ;--
                          01AF
                          01AF    FAC_BRO:                                        ; Special processing of BRO
      F4 67    24    E1   01AF            BBC     #DAP$V_GEQ_V56,(R7),SUC   ; Branch if partner uses DAP before V5.6
      F0 67    34    E0   01B3            BBS     #DAP$V_VAXVMS,(R7),SUC    ; Branch if partner is VAX/VMS
   00   00C6   C7    91   01B7            CMPB    NWA$B_ORG(R7),#NWA$K_SEQ; Branch if SEQ organization
               E9    13   01BC            BEQL    SUC                       ;  else fall thru if REL or IDX
                          01BE
                          01BE    ;+
                          01BE    ; Build and send DAP Access Complete message to partner.
                          01BE    ;-
                          01BE
                          01BE    BRO_SEND_CMP:                             ;
                          01BE            $SETBIT #NWA$V_LAST_MSG,(R7)      ; Declare this last message to block
         50    07    D0   01C2            MOVL    #DAP$K_CMP_MSG,R0         ; Get message type value
            FE38'    30   01C5            BSBW    NT$BUILD_HEAD             ; Construct message header
         85    01    90   01C8            MOVB    #DAP$K_CLOSE,(R5)+        ; Store CMPFUNC field
            FE32'    30   01CB            BSBW    NT$BUILD_TAIL             ; Finish building message
            FE2F'    30   01CE            BSBW    NT$TRANSMIT               ; Send Access Complete message to FAL
         DA    50    E9   01D1            BLBC    R0,FAIL3                  ; Branch on failure
                          01D4
                          01D4    ;+
                          01D4    ; Receive DAP Access Complete message from partner.
                          01D4    ;-
                          01D4
                          01D4    BRO_RECV_CMP:                             ;
                          01D4            $SETBIT #DAP$K_CMP_MSG,DAP$L_MSG_MASK(R7)
                          01D9                                              ; Expect response of Access Complete msg
            FE24'    30   01D9            BSBW    NT$RECEIVE                ; Get reply from FAL
         CF    50    E9   01DC            BLBC    R0,FAIL3                  ; Branch on failure
                          01DF
                          01DF    ;+
                          01DF    ; Now reopen the file in block I/O mode.
                          01DF    ;-
                          01DF
                          01DF            $CLRBIT #IFB$V_BRO,IFB$B_FAC(R9); Transform BRO request into a BIO
                          01E4            $SETBIT #IFB$V_BIO,IFB$B_FAC(R9);  request
               01    B0   01E9            MOVW    #DAP$M_DSP_ATT,-          ; Do not request return of XAB info
            00D0 C7       01EB                    NWA$W_DISPLAY(R7)         ;  as we already have it from first open
            FE59     31   01EE            BRW     SEND_ATT                  ; Reopen the file in  block I/O mode
```

```
                                01F1            .SBTTL  NT$RECV_EXT_ATT
                                01F1
                                01F1    ;++
                                01F1    ; This routine receives and decodes DAP Extended Attributes messages from
                                01F1    ; partner and updates the user Allocation, Date and Time, Key Definition,
                                01F1    ; Protection, Revision Date and Time, and Summary XABs as appropriate.
                                01F1    ;
                                01F1    ; A mask (NWA$L_MSG_MASK) is used to determine if all requested Extended
                                01F1    ; Attributes messages (NWA$W_DISPLAY) have been received before allowing a
                                01F1    ; DAP NAM or ACK message to be received and processed. NT$DECODE_xxx routines
                                01F1    ; each clear their respective mask bit after processing a DAP message.
                                01F1    ;
                                01F1    ; Note: For indexed files, multiple Allocation and Key Definition messages
                                01F1    ;       may be returned.
                                01F1    ;-
                                01F1
                                01F1    NT$RECV_EXT_ATT::                               ; Entry point
         51    00D0 C7    3C    01F1            MOVZWL  NWA$W_DISPLAY(R7),R1        ; Get DAP message request mask
                     52    D4    01F6            CLRL    R2                          ; Clear valid messages to receive mask
                                01F8            $MAPBIT DAP$V_DSP_ALL,DAP$K_ALL_MSG; Map request for Allocation msg
                                0200            $MAPBIT DAP$V_DSP_KEY,DAP$K_KEY_MSG; Map request for Key Definition msg
                                0208            $MAPBIT DAP$V_DSP_PRO,DAP$K_PRO_MSG; Map request for Protection msg
                                0210            $MAPBIT DAP$V_DSP_SUM,DAP$K_SUM_MSG; Map request for Summary msg
                                0218            $MAPBIT DAP$V_DSP_TIM,DAP$K_TIM_MSG; Map request for Date and Time msg
       00D4 C7    52    D0    0220            MOVL    R2,NWA$L_MSG_MASK(R7)       ; Save valid message mask for use again
            00D4 C7    D0    0225    LOOP:   MOVL    NWA$L_MSG_MASK(R7),-        ; Expect response of any of these DAP
               1C A7          0229                    DAP$L_MSG_MASK(R7)          ;  messages
                  58    13    022B            BEQL    DONE                        ; Branch if no more to receive
               FDD0'    30    022D    LOOP1:  BSBW    NT$RECEIVE                  ; Get reply from FAL
               55 50    E9    0230            BLBC    R0,FAIL                     ; Branch on failure
                                0233
                                0233            ASSUME  DAP$K_KEY_MSG EQ 10
                                0233            ASSUME  DAP$K_ALL_MSG EQ 11
                                0233            ASSUME  DAP$K_SUM_MSG EQ 12
                                0233            ASSUME  DAP$K_TIM_MSG EQ 13
                                0233            ASSUME  DAP$K_PRO_MSG EQ 14
                                0233
               45'AF    9F    0233            PUSHAB  B^LOOP2                     ; Push return address on stack
                                0236            $CASEB  SELECTOR=DAP$B_TYPE(R7)-; Dispatch to process message:
                                0236                    BASE=#DAP$K_KEY_MSG-      ;
                                0236                    DISPL=<-                  ;
                                0236                        NT$DECODE_KEY-        ;  Key Definition message
                                0236                        NT$DECODE_ALL-        ;  Allocation message
                                0236                        NT$DECODE_SUM-        ;  Summary message
                                0236                        NT$DECODE_TIM-        ;  Date and Time message
                                0236                        NT$DECODE_PRO-        ;  Protection message
                                0236                    >                         ;
                                0245
                                0245    ;+
                                0245    ; If this is an indexed file and an Allocation or Key Definition message has just
                                0245    ; been processed, look ahead to see if the next message is of the same type.
                                0245    ; If so, process it; otherwise don't allow any more of this type (i.e.,
                                0245    ; multiple Allocation and Key Definition messages must be received as a block).
                                0245    ;-
                                0245
         20    00C6 C7    91    0245    LOOP2:  CMPB    NWA$B_ORG(R7),#NWA$K_IDX; Branch if not IDX organization
                  D9    12    024A            BNEQ    LOOP                        ;
```

```
                    024C               $CASEB   SELECTOR=DAP$B_TYPE(R7)- ; Message just processed was:
                    024C                        BASE=#DAP$K_KEY_MSG-     ;
                    024C                        DISPL=<-                 ;
                    024C                             DECODE_NXT_KEY-     ;  Key Definition message
                    024C                             DECODE_NXT_ALL-     ;  Allocation message
                    024C                        >                        ;
         CE    11   0255               BRB      LOOP                     ;  all others
                    0257   DECODE_NXT_KEY:                               ; Handle multiple KEY messages
                    0257               $SETBIT  #NWA$V_NODECODE,(R7)     ; Read next DAP message (if required)
      FDA2'    30   025B               BSBW     NT$RECEIVE               ;  but don't parse it
      27 50    E9   025E               BLBC     R0,FAIL                  ; Branch on failure
      0C B7    91   0261               CMPB     @DAP$Q_MSG_BUF1+4(R7),-  ; Branch if this is not another
         0A         0264                        #DAP$K_KEY_MSG           ;  Key Definition message
         BE    12   0265               BNEQ     LOOP                     ;  (disallow any more KEY messages)
                    0267               $SETBIT  #DAP$K_KEY_MSG,DAP$L_MSG_MASK(R7)
         BF    11   026C               BRB      LOOP1                    ; Process this Key Definition message
                    026E   DECODE_NXT_ALL:                               ; Handle multiple Allocation messages
                    026E               $SETBIT  #NWA$V_NODECODE,(R7)     ; Read next DAP message (if required)
      FD8B'    30   0272               BSBW     NT$RECEIVE               ;  but don't parse it
      10 50    E9   0275               BLBC     R0,FAIL                  ; Branch on failure
      0C B7    91   0278               CMPB     @DAP$Q_MSG_BUF1+4(R7),-  ; Branch if this is not another
         0B         027B                        #DAP$K_ALL_MSG           ;  Allocation message
         A7    12   027C               BNEQ     LOOP                     ;  (disallow any more ALL messages)
                    027E               $SETBIT  #DAP$K_ALL_MSG,DAP$L_MSG_MASK(R7)
         A8    11   0283               BRB      LOOP1                    ; Process this Allocation message
                    0285   DONE:       RMSSUC                            ; Return success
               05   0288   FAIL:       RSB                               ; FAIL with RMS code in R0
```

```
                          0289          .SBTTL  OPEN_UPDATE_FAB
                          0289
                          0289  ;++
                          0289  ; Update the user FAB from the Attributes message.
                          0289  ;
                          0289  ; Note: BLS and MRN will be updated directly in the FAB, whereas, the other
                          0289  ;       fields will be updated in the IFB and then returned to the FAB by the
                          0289  ;       RMS0OPEN exit code.
                          0289  ;--
                          0289
                          0289  OPEN_UPDATE_FAB:                                ; Entry point
                          0289
                          0289  ;
                          0289  ; Process the DAP ORG, MRN, BLS, RFM, and RAT fields.
                          0289  ;
                          0289
       04   04        EF  0289          EXTZV   #IFB$V_ORG,#IFB$S_ORG,-  ; Note that DAP$B_ORG is in same
    51    45 A7           028C                  DAP$B_ORG(R7),R1         ;  format as IFB$B_RFMORG
    23 A9    51       90  028F          MOVB    R1,IFB$B_ORGCASE(R9)     ; Store file organization
       00    51       91  0293          CMPB    R1,#DAP$K_SEQ            ; Branch if SEQ organization
             0D       13  0296          BEQL    10$
    10    45 A7       91  0298          CMPB    DAP$B_ORG(R7),#DAP$K_REL; Branch if not REL organization
             0C       12  029C          BNEQ    20$                      ;
 38 A8    58 A7       D0  029E          MOVL    DAP$L_MRN(R7),FAB$L_MRN(R8)
             05       11  02A3          BRB     20$
 3C A8    48 A7       B0  02A5  10$:    MOVW    DAP$W_BLS(R7),FAB$W_BLS(R8)
 50 A9    46 A7       90  02AA  20$:    MOVB    DAP$B_RFM(R7),IFB$B_RFMORG(R9)
            00B4      30  02AF          BSBW    NT$MOD_RAT               ; Modify RAT bits returned from FAL
                          02B2                                           ;  as required
 51 A9    47 A7       90  02B2          MOVB    DAP$B_RAT(R7),IFB$B_RAT(R9)
                          02B7
                          02B7  ;
                          02B7  ; Process the DAP MRS and LRL fields.
                          02B7  ;
                          02B7
 60 A9    4A A7       B0  02B7          MOVW    DAP$W_MRS(R7),IFB$W_MRS(R9)
 52 A9    70 A7       B0  02BC          MOVW    DAP$W_LRL(R7),IFB$W_LRL(R9)
             0B       12  02C1          BNEQ    30$                      ; Branch if non-zero
    01    46 A7       91  02C3          CMPB    DAP$B_RFM(R7),#DAP$K_FIX; Branch if record format is not
             05       12  02C7          BNEQ    30$                      ;  fixed length
 52 A9    4A A7       B0  02C9          MOVW    DAP$W_MRS(R7),IFB$W_LRL(R9)
                          02CE
                          02CE  ;
                          02CE  ; Process the DAP ALQ and HBK fields.
                          02CE  ;
                          02CE  ; Note: ALQ and HBK are equivalent, but not all non-VAX nodes return HBK.
                          02CE  ;
                          02CE
 70 A9    4C A7       D0  02CE  30$:    MOVL    DAP$L_ALQ1(R7),IFB$L_HBK(R9)
                          02D3
                          02D3  ;
                          02D3  ; Process the DAP BKS, FSZ, and DEQ fields.
                          02D3  ;
                          02D3
 5E A9    50 A7       90  02D3  40$:    MOVB    DAP$B_BKS(R7),IFB$B_BKS(R9)
 5F A9    51 A7       90  02D8          MOVB    DAP$B_FSZ(R7),IFB$B_FSZ(R9)
 62 A9    54 A7       B0  02DD          MOVW    DAP$W_DEQ1(R7),IFB$W_DEQ(R9)
```

```
                              02E2
                              02E2   ;
                              02E2   ; Process the DAP FOP field.
                              02E2   ;
                              02E2
          51    64 A7    D0   02E2          MOVL     DAP$L_FOP1(R7),R1          ; Get DAP FOP bits
                   52    D4   02E6          CLRL     R2                         ; Clear resultant FOP bits
                              02E8          $MAPBIT DAP$V_CTG,FAB$V_CTG         ; Map CTG bit
                              02F0          $MAPBIT DAP$V_CBT,FAB$V_CBT         ; Map CBT bit
                              02F8          $MAPBIT DAP$V_RCK,FAB$V_RCK         ; Map RCK bit
                              0300          $MAPBIT DAP$V_WCK,FAB$V_WCK         ; Map WCK bit
    04 A8    00B00200 8F   CA   0308          BICL2    #<<FAB$M_CTG>!-            ; Clear FOP bits in user FAB
                              0310                     <FAB$M_CBT>!-             ;  that may be updated
                              0310                     <FAB$M_RCK>!-             ;
                              0310                     <FAB$M_WCK>!-             ;
                              0310                     0>,FAB$L_FOP(R8)          ;
          04 A8    52    C8   0310          BISL2    R2,FAB$L_FOP(R8)           ; Update FOP field
                   05         0314          RSB                                 ; Exit
```

```
                                 0315         .SBTTL  NT$UPDATE_FHC - UPDATE FHC XAB
                                 0315
                                 0315   ;++
                                 0315   ; Update the user File Header Characteristics XAB from the Attributes message.
                                 0315   ;--
                                 0315
                                 0315   NT$UPDATE_FHC::                              ; Entry point
          56    0108 C7    D0    0315           MOVL    NWA$L_FHCXABADR(R7),R6   ; Get address of user FHCXAB
                      49    13    031A           BEQL    30$                      ; Branch if none
                                 031C
                                 031C   ;
                                 031C   ; Process the DAP RAT, BKS, DEQ, EBK, FFB, FSZ, and SBN fields.
                                 031C   ;
                                 031C
                    0047    30    031C           BSBW    NT$MOD_RAT               ; Modify RAT bits returned from FAL
                                 031F                                            ;  as required
       09 A6     47 A7    90    031F           MOVB    DAP$B_RAT(R7),XAB$B_ATR(R6)
       16 A6     50 A7    90    0324           MOVB    DAP$B_BKS(R7),XAB$B_BKZ(R6)
       1A A6     54 A7    B0    0329           MOVW    DAP$W_DEQ1(R7),XAB$W_DXQ(R6)
       10 A6     78 A7    D0    032E           MOVL    DAP$L_EBK(R7),XAB$L_EBK(R6)
       14 A6     72 A7    B0    0333           MOVW    DAP$W_FFB(R7),XAB$W_FFB(R6)
       17 A6     51 A7    90    0338           MOVB    DAP$B_FSZ(R7),XAB$B_HSZ(R6)
       28 A6     7C A7    D0    033D           MOVL    DAP$L_SBN(R7),XAB$L_SBN(R6)
                                 0342
                                 0342   ;
                                 0342   ; Process the DAP MRS and LRL fields.
                                 0342   ;
                                 0342
       18 A6     4A A7    B0    0342           MOVW    DAP$W_MRS(R7),XAB$W_MRZ(R6)
       0A A6     70 A7    B0    0347           MOVW    DAP$W_LRL(R7),XAB$W_LRL(R6)
                      08    12    034C           BNEQ    10$                      ; Branch if non-zero
          01     46 A7    91    034E           CMPB    DAP$B_RFM(R7),#DAP$K_FIX; Branch if format is not
                      05    12    0352           BNEQ    10$                      ;  fixed length
       0A A6     4A A7    B0    0354           MOVW    DAP$W_MRS(R7),XAB$W_LRL(R6)
                                 0359
                                 0359   ;
                                 0359   ; Process the DAP ALQ and HBK fields.
                                 0359   ;
                                 0359   ; Note: ALQ and HBK are equivalent, but not all non-VAX nodes return HBK.
                                 0359   ;
                                 0359
       0C A6     4C A7    D0    0359   10$:    MOVL    DAP$L_ALQ1(R7),XAB$L_HBK(R6)
                                 035E
                                 035E   ;
                                 035E   ; Process the DAP RFM and ORG fields which are combined into one XAB RFO field.
                                 035E   ;
                                 035E
                                 035E           ASSUME  DAP$K_UDF EQ FAB$C_UDF
                                 035E           ASSUME  DAP$K_FIX EQ FAB$C_FIX
                                 035E           ASSUME  DAP$K_VAR EQ FAB$C_VAR
                                 035E           ASSUME  DAP$K_VFC EQ FAB$C_VFC
                                 035E           ASSUME  DAP$K_STM EQ FAB$C_STM
                                 035E           ASSUME  DAP$K_STMLF EQ FAB$C_STMLF
                                 035E           ASSUME  DAP$K_STMCR EQ FAB$C_STMCR
                                 035E
 08 A6   45 A7   46 A7    81    035E   20$:    ADDB3   DAP$B_RFM(R7),DAP$B_ORG(R7),XAB$B_RFO(R6)
                            05    0365   30$:    RSB                              ; Exit
```

```
NTO
Sym

FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAB
FAC
FAI
FAI
FAI
FAI
FAI
FIL
FWA
FWA
IFE
IFE
IFE
IFE
IFE
IFE
IFE
IFE
IFE
IFE
IFE
IFE
IFE
IFE
IFE
```

```
                    0366       .SBTTL  NT$MOD_RAT
                    0366
                    0366 ;++
                    0366 ; This routine converts the DAP RAT field received from the remote FAL to a
                    0366 ; form that may be retruned to the user. In particular, the DAP$V_EMBEDDED bit
                    0366 ; is cleared, and if the DAP RFM field = STM with no other carriage control
                    0366 ; bits set, RAT is forced to CR.
                    0366 ;--
                    0366
                    0366 NT$MOD_RAT::                                ; Entry point
                    0366         $CLRBIT #DAP$V_EMBEDDED,DAP$B_RAT(R7) ; Discard the embedded bit
04    46 A7    91   036B         CMPB    DAP$B_RFM(R7),#DAP$K_STM; Branch if not STM format
         0B    12   036F         BNEQ    10$                     ;
47 A7    07    93   0371         BITB    #<<DAP$M_FTN>!-         ; If RAT = embedded or none for STM
                    0375                   <DAP$M_CR>!-          ;  format file ...
                    0375                   <DAP$M_PRN>!-         ;
                    0375                   0>,DAP$B_RAT(R7)      ;
         05    12   0375         BNEQ    10$                     ;
                    0377         $SETBIT #DAP$V_CR,DAP$B_RAT(R7) ; Force implied carriage control
               05   037C 10$:    RSB                             ; Exit
```

```
NTO
Sym

NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
NWA
OPE
PAR
PAR
PAR
PAR
PIC
REC
REC
REC
REC
SEN
SEN
SUC
TP1
```

```
                          037D          .SBTTL  NT$DECODE_KEY - UPDATE KEY XAB
                          037D
                          037D  ;++
                          037D  ; A Key Definition message has been received and decoded in the DAP control
                          037D  ; Block. Update the next user Key Definition XAB in chain.
                          037D  ;
                          037D  ; Note: Multiple Key Definition XABs are valid only for indexed files.
                          037D  ;--
                          037D
                          037D  NT$DECODE_KEY::                              ; Entry point
         011D C7     95   037D          TSTB    NWA$B_KEYXABCNT(R7)          ; Branch if there are more KEYXABs
           03        12   0381          BNEQ    10$                          ;  in chain
         00AE        31   0383          BRW     30$                          ; Branch if none
  56     010C C7     D0   0386  10$:    MOVL    NWA$L_KEYXABADR(R7),R6       ; Get address of next KEYXAB in chain
                          038B
                          038B  ;
                          038B  ; Process the DAP FLG field.
                          038B  ;
                          038B
    51     48 A7     9A   038B          MOVZBL  DAP$B_FLG(R7),R1             ; Get DAP FLG bits
             52      D4   038F          CLRL    R2                           ; Clear RMS FLG bits
                          0391          $MAPBIT DAP$V_DUP,XAB$V_DUP          ; Map DUP bit
                          0399          $MAPBIT DAP$V_CHG,XAB$V_CHG          ; Map CHG bit
                          03A1          $MAPBIT DAP$V_NUL_CHR,XAB$V_NUL      ; Map NUL bit
   12 A6     52      90   03A9          MOVB    R2,XAB$B_FLG(R6)             ; Update FLG field in XAB
                          03AD
                          03AD  ;
                          03AD  ; Process the DAP DFL, IFL, REF, NUL, IAN, LAN, DAN, DTP, RVB, DVB,
                          03AD  ; DBS, IBS, LVL, TKS, and MRL fields.
                          03AD  ;
                          03AD
1C A6     44 A7      90   03AD          MOVB    DAP$W_DFL(R7),XAB$W_DFL(R6)
1A A6     46 A7      90   03B2          MOVB    DAP$W_IFL(R7),XAB$W_IFL(R6)
17 A6     6C A7      90   03B7          MOVB    DAP$B_REF(R7),XAB$B_REF(R6)
15 A6     6D A7      90   03BC          MOVB    DAP$B_NUL(R7),XAB$B_NUL(R6)
08 A6     6E A7      90   03C1          MOVB    DAP$B_IAN(R7),XAB$B_IAN(R6)
09 A6     6F A7      90   03C6          MOVB    DAP$B_LAN(R7),XAB$B_LAN(R6)
0A A6     70 A7      90   03CB          MOVB    DAP$B_DAN(R7),XAB$B_DAN(R6)
13 A6     71 A7      90   03D0          MOVB    DAP$B_DTP(R7),XAB$B_DTP(R6)
0E A6     74 A7      D0   03D5          MOVL    DAP$L_RVB(R7),XAB$L_RVB(R6)
3C A6     78 A7      D0   03DA          MOVL    DAP$L_DVB(R7),XAB$L_DVB(R6)
0D A6     7C A7      90   03DF          MOVB    DAP$B_DBS(R7),XAB$B_DBS(R6)
0C A6     7D A7      90   03E4          MOVB    DAP$B_IBS(R7),XAB$B_IBS(R6)
0B A6     7E A7      90   03E9          MOVB    DAP$B_LVL(R7),XAB$B_LVL(R6)
16 A6     7F A7      90   03EE          MOVB    DAP$B_TKS(R7),XAB$B_TKS(R6)
18 A6     72 A7      B0   03F3          MOVW    DAP$W_MRL(R7),XAB$W_MRL(R6)
                          03F8
                          03F8  ;
                          03F8  ; Process the DAP NSG, [illegible]S, and SIZ fields.
                          03F8  ;
                          03F8  ; Note: NT$DECODE_MSG guarantees that 0 < DAP$B_NSG < 9.
                          03F8  ;
                          03F8
             58      DD   03F8          PUSHL   R8                           ; Save register
    58     49 A7     9A   03FA          MOVZBL  DAP$B_NSG(R7),R8             ; Get # key segments
   14 A6     58      90   03FE          MOVB    R8,XAB$B_NSG(R6)             ; Update NSG field in XAB
   5C A7     58      28   0402          MOVC3   R8,DAP$B_SIZ(R7),-           ; Copy 1 to 8 key size values
```

```
               2E A6        0406                       XAB$B_SIZ(R6)                ;  to XAB
      58   58  01    78     0408             ASHL      #1,R8,R8                     ; Double byte count
        4C A7  58    28     040C             MOVC3     R8,DAP$W_POS(R7),-           ; Copy 1 to 8 key position values
               1E A6        0410                       XAB$W_POS(R6)                ;  to XAB
               58 8ED0      0412             POPL      R8                           ; Restore register
                            0415
                            0415      ;
                            0415      ; Process the DAP KNM field.
                            0415      ;
                            0415
          55   38 A6  D0    0415             MOVL      XAB$L_KNM(R6),R5             ; Get address of key name buffer
                  0F  13    0419             BEQL      20$                          ; Branch if no buffer supplied
  65   20    0A A9    0D    041B             PROBEW    IFB$B_MODE(R9),#32,(R5)      ; Test writeability
                  08  13    0420             BEQL      20$                          ; Branch on failure
              64 A7   2C    0422             MOVC5     DAP$Q_KNM(R7),-              ; Copy DAP key name string
              68 B7         0425                       @DAP$Q_KNM+4(R7),-           ;  to 32 byte XAB buffer
      65   20    00         0427                       #0,#32,(R5)                  ;
                            042A
                            042A      ;
                            042A      ; Set-up for next time thru.
                            042A      ;
                            042A
         011D C7   97       042A      20$:   DECB      NWA$B_KEYXABCNT(R7)          ; Reduce count of KEYXABs left
           04 A6   D0       042E             MOVL      XAB$L_NXT(R6),-              ; Save address of next KEYXAB in chain
         010C C7            0431                       NWA$L_KEYXABADR(R7)          ;  (valid only if NWA$B_KEYXABCNT > 0)
                            0434      30$:   $CLRBIT   #DAP$K_KEY_MSG,NWA$L_MSG_MASK(R7); Check it off from list
                   05       043A             RSB                                    ; Process next DAP message
```

```
                              043B          .SBTTL  NT$DECODE_ALL - UPDATE ALL XAB
                              043B
                              043B  ;++
                              043B  ; An Allocation message has been received and decoded in the DAP control block.
                              043B  ; Update the next user Allocation XAB in chain.
                              043B  ;
                              043B  ; Note: Multiple Allocation XABs are valid only for indexed files.
                              043B  ;--
                              043B
                              043B  NT$DECODE_ALL::                                  ; Entry point
          011C C7     95      043B          TSTB    NWA$B_ALLXABCNT(R7)      ; Branch if there are more ALLXABs
               03     12      043F          BNEQ    10$                      ;  in chain
             0093     31      0441          BRW     60$                      ; Branch aid
   56     0100 C7     D0      0444  10$:    MOVL    NWA$L_ALLXABADR(R7),R6   ; Get address of next ALLXAB in chain
                              0449
                              0449  ;
                              0449  ; Process the DAP ALN field.
                              0449  ;
                              0449
                              0449          ASSUME  DAP$K_ANY EQ 0
                              0449          ASSUME  DAP$K_CYL EQ XAB$C_CYL
                              0449          ASSUME  DAP$K_LBN EQ XAB$C_LBN
                              0449          ASSUME  DAP$K_VBN EQ XAB$C_VBN
                              0449
   09 A6     44 A7    90      0449          MOVB    DAP$B_ALN(R7),XAB$B_ALN(R6)
                              044E
                              044E  ;
                              044E  ; Process the DAP AOP field.
                              044E  ;
                              044E
      51     45 A7    9A      044E          MOVZBL  DAP$B_AOP(R7),R1         ; Get DAP AOP bits
                52    D4      0452          CLRL    R2                       ; Clear RMS AOP bits
                              0454          $MAPBIT DAP$V_HRD,XAB$V_HRD      ; Map HRD bit
                              045C          $MAPBIT DAP$V_CBT2,XAB$V_CBT     ; Map CBT bit
                              0464          $MAPBIT DAP$V_CTG2,XAB$V_CTG     ; Map CTG bit
                              046C          $MAPBIT DAP$V_ONC,XAB$V_ONC      ; Map ONC bit
   08 A6        52    90      0474          MOVB    R2,XAB$B_AOP(R6)         ; Update AOP field in XAB
                              0478
                              0478  ;
                              0478  ; Process the DAP VOL, LOC, ALQ, AID, BKZ, and DEQ fields.
                              0478  ;
                              0478
   0A A6     42 A7    B0      0478          MOVW    DAP$W_VOL(R7),XAB$W_VOL(R6)
   0C A6     48 A7    D0      047D          MOVL    DAP$L_LOC(R7),XAB$L_LOC(R6)
   10 A6     4C A7    D0      0482          MOVL    DAP$L_ALQ2(R7),XAB$L_ALQ(R6)
   17 A6     50 A7    90      0487          MOVB    DAP$B_AID(R7),XAB$B_AID(R6)
   16 A6     51 A7    90      048C          MOVB    DAP$B_BKZ(R7),XAB$B_BKZ(R6)
   14 A6     52 A7    B0      0491          MOVW    DAP$W_DEQ2(R7),XAB$W_DEQ(R6)
                              0496
                              0496  ;
                              0496  ; If the DAP ALQ, BKZ, DEQ, or AOP fields are not explicitly returned in the
                              0496  ; Allocation message, they will be defaulted to values received in corresponding
                              0496  ; fields of the Attributes message.
                              0496  ;
                              0496
      51     40 A7    3C      0496          MOVZWL  DAP$W_ALLMENU(R7),R1     ; Get allocation menu field
   05 51        05    E0      049A          BBS     #DAP$V_ALQ2,R1,20$       ; Ok if explicit value returned
```

```
        70 A9   D0  049E             MOVL     IFB$L_HBK(R9),-            ; Default ALQ value
        10 A6       04A1                      XAB$L_ALQ(R6)               ;
  05 51    07   E0  04A3   20$:      BBS      #DAP$V_BKZ,R1,30$           ; Ok if explicit value returned
        5E A9   90  04A7             MOVB     IFB$B_BKS(R9),-            ; Default BKZ value
        16 A6       04AA                      XAB$B_BKZ(R6)               ;
  05 51    08   E0  04AC   30$:      BBS      #DAP$V_DEQ2,R1,40$          ; Ok if explicit value returned
        62 A9   B0  04B0             MOVW     IFB$W_DEQ(R9),-            ; Default DEQ value
        14 A6       04B3                      XAB$W_DEQ(R6)               ;
     14 51 02   E0  04B5   40$:      BBS      #DAP$V_AOP,R1,50$           ; Ok if explicit value returned
05 04 A8   15   E1  04B9             BBC      #FAB$V_CBT,FAB$L_FOP(R8),45$
                    04BE             $SETBIT  #XAB$V_CBT,XAB$B_AOP(R6); Map CBT bit
05 04 A8   14   E1  04C3   45$:      BBC      #FAB$V_CTG,FAB$L_FOP(R8),50$
                    04C8             $SETBIT  #XAB$V_CTG,XAB$B_AOP(R6); Map CTG bit
                    04CD
                    04CD       ;
                    04CD       ; Set-up for next time thru.
                    04CD       ;
                    04CD
     011C C7    97  04CD   50$:      DECB     NWA$B_ALLXABCNT(R7)        ; Reduce count of Allocation XABs left
        04 A6   D0  04D1             MOVL     XAB$L_NXT(R6),-            ; Save address of next ALLXAB in chain
     0100 C7        04D4                      NWA$L_ALLXABADR(R7)        ;  (valid only if NWA$B_ALLXABCNT > 0)
                    04D7   60$:      $CLRBIT  #DAP$K_ALL_MSG,NWA$L_MSG_MASK(R7); Check it off from list
           05       04DD             RSB                                 ; Process next DAP message
```

```
                                04DE          .SBTTL  NT$DECODE_ALL_A - UPDATE ALL XAB
                                04DE
                                04DE    ;++
                                04DE    ; An Attributes message has been received and decoded in the DAP control block.
                                04DE    ; Update the user Allocation XAB from the Attributes message (in lieu of the
                                04DE    ; Allocation message) because the remote FAL does not support the Allocation
                                04DE    ; message.
                                04DE    ;--
                                04DE
                                04DE   NT$DECODE_ALL_A::                              ; Entry point
      56    0100 C7     D0      04DE          MOVL    NWA$L_ALLXABADR(R7),R6  ; Get address of user ALLXAB
                 3C     13      04E3          BEQL    10$                      ; Branch if none
      20    00C6 C7     91      04E5          CMPB    NWA$B_ORG(R7),#NWA$K_IDX; Do not update user ALLXAB
                 35     13      04EA          BEQL    10$                      ;   if ORG = IDX
                                04EC
                                04EC    ;
                                04EC    ; Process the DAP FOP field.
                                04EC    ;
                                04EC    ; Note: The HRD and ONC bits are not mapped into the user AOP field because
                                04EC    ;       the FOP field does not contain these.
                                04EC    ;
                                04EC
       51     64 A7     D0      04EC          MOVL    DAP$L_FOP1(R7),R1        ; Get DAP FOP bits
                 52     D4      04F0          CLRL    R2                       ; Clear resultant AOP bits
                                04F2          $MAPBIT DAP$V_CTG,XAB$V_CTG      ; Map CTG bit
                                04FA          $MAPBIT DAP$V_CBT,XAB$V_CBT      ; Map CBT bit
    08 A6         52    90      0502          MOVB    R2,XAB$B_AOP(R6)         ; Update AOP field in XAB
                                0506
                                0506    ;
                                0506    ; Process the DAP ALQ, BKS, and DEQ fields.
                                0506    ;
                                0506
 10 A6     4C A7        D0      0506          MOVL    DAP$L_ALQ1(R7),XAB$L_ALQ(R6)
 16 A6     50 A7        90      050B          MOVB    DAP$B_BKS(R7),XAB$B_BKZ(R6)
 14 A6     54 A7        B0      0510          MOVW    DAP$W_DEQ1(R7),XAB$W_DEQ(R6)
                                0515
                                0515    ;
                                0515    ; Zero the XAB ALN, LOC, AID, and VOL fields because these are not obtainable
                                0515    ; from the Attributes message.
                                0515    ;
                                0515
          09 A6         94      0515          CLRB    XAB$B_ALN(R6)            ; Zero these fields
          0C A6         D4      0518          CLRL    XAB$L_LOC(R6)            ;
          17 A6         94      051B          CLRB    XAB$B_AID(R6)            ;
          0A A6         B4      051E          CLRW    XAB$W_VOL(R6)            ;
                 05             0521   10$:   RSB                              ; Exit
```

```
                                0522        .SBTTL  NT$DECODE_SUM - UPDATE SUM XAB
                                0522
                                0522  ;++
                                0522  ; A Summary message has been received abd decoded in the DAP control block.
                                0522  ; Update the user Summary XAB.
                                0522  ;--
                                0522
                                0522  NT$DECODE_SUM::                                 ; Entry point
            56   0118 C7   D0   0522        MOVL    NWA$L_SUMXABADR(R7),R6  ; Get address of user SUMXAB
                       0F   13  0527        BEQL    10$                     ; Branch if none
                                0529
                                0529  ;
                                0529  ; Process the DAP NOK, NOA, and PRV fields.
                                0529  ;
                                0529
          09 A6    44 A7   90   0529        MOVB    DAP$B_NOK(R7),XAB$B_NOK(R6)
          08 A6    45 A7   90   052E        MOVB    DAP$B_NOA(R7),XAB$B_NOA(R6)
          0A A6    42 A7   B0   0533        MOVW    DAP$W_PVN(R7),XAB$W_PVN(R6)
                                0538  10$:  $CLRBIT #DAP$K_SUM_MSG,NWA$L_MSG_MASK(R7); Check it off from list
                           05   053E        RSB                             ; Process next DAP message
```

```
                          053F              .SBTTL  NT$DECODE_TIM - UPDATE DAT XAB
                          053F
                          053F      ;++
                          053F      ; A Date and Time message has been received and decoded in the DAP control
                          053F      ; block. Update both the user Date and Time XAB and the Revision Date and Time
                          053F      ; XAB as appropriate.
                          053F      ;--
                          053F
                          053F      NT$DECODE_TIM::                                   ; Entry point
                          053F
                          053F      ;
                          053F      ; First update the Date and Time XAB if present.
                          053F      ;
                          053F
 56    0104 C7      D0    053F              MOVL    NWA$L_DATXABADR(R7),R6  ; Get address of user DATXAB
              1F    13    0544              BEQL    10$                     ; Branch if none
                          0546
                          0546      ;
                          0546      ; Process the DAP CDT, RDT, EDT, BDT, and RVN fields.
                          0546      ;
                          0546
          48 A7     7D    0546              MOVQ    DAP$Q_CDT(R7),-         ; Copy creation date and time
          14 A6           0549                      XAB$Q_CDT(R6)           ;  binary value to XAB
          50 A7     7D    054B              MOVQ    DAP$Q_RDT(R7),-         ; Copy revision date and time
          0C A6           054E                      XAB$Q_RDT(R6)           ;  binary value to XAB
          58 A7     7D    0550              MOVQ    DAP$Q_EDT(R7),-         ; Copy expiration date and time
          1C A6           0553                      XAB$Q_EDT(R6)           ;  binary value to XAB
          42 A7     B0    0555              MOVW    DAP$W_RVN(R7),-         ; Store revision number value in XAB
          08 A6           0558                      XAB$W_RVN(R6)           ;
          01 A6     91    055A              CMPB    XAB$B_BLN(R6),-         ; Branch if length of XAB is too small
             2C           055D                      #XAB$C_DATLEN           ;  to contain BDT field (i.e., it's a
             05     1F    055E              BLSSU   10$                     ;  V2 length XAB)
          60 A7     7D    0560              MOVQ    DAP$Q_BDT(R7),-         ; Copy backup date and time
          24 A6           0563                      XAB$Q_BDT(R6)           ;  binary value to XAB
                          0565
                          0565      ;
                          0565      ; Next update the Revision Date and Time XAB if present.
                          0565      ;
                          0565
 56    0114 C7      D0    0565      10$:    MOVL    NWA$L_RDTXABADR(R7),R6  ; Get address of user RDTXAB
              0A    13    056A              BEQL    20$                     ; Branch if none
                          056C
                          056C      ;
                          056C      ; Process the DAP RDT and RVN fields again.
                          056C      ;
                          056C
          50 A7     7D    056C              MOVQ    DAP$Q_RDT(R7),-         ; Copy revision date and time
          0C A6           056F                      XAB$Q_RDT(R6)           ;  binary value to XAB
          42 A7     B0    0571              MOVW    DAP$W_RVN(R7),-         ; Store revision number value in XAB
          08 A6           0574                      XAB$W_RVN(R6)           ;
                          0576
                          0576      20$:    $CLRBIT #DAP$K_TIM_MSG,NWA$L_MSG_MASK(R7); Check it off from list
                    05    057C              RSB                             ; Process next DAP message
```

```
                        057D          .SBTTL  NT$DECODE_PRO - UPDATE PRO XAB
                        057D
                        057D    ;++
                        057D    ; A Protection message has been received and decoded in the DAP control block.
                        057D    ; Update the user Protection XAB.
                        057D    ;--
                        057D
                        057D    NT$DECODE_PRO::                                     ; Entry point
        56    0110 C7   D0  057D          MOVL    NWA$L_PROXABADR(R7),R6    ; Get address of user PROXAB
                   75   13  0582          BEQL    40$                       ; Branch if none
              0C   A6   D4  0584          CLRL    XAB$L_UIC(R6)             ; Set UIC to default value
                        0587
                        0587    ;
                        0587    ; Process the DAP OWNER field.
                        0587    ;
                        0587
        54    48   A7   7D  0587          MOVQ    DAP$Q_OWNER(R7),R4        ; Get descriptor of ASCII string
        5B 8F      65   91  058B          CMPB    (R5),#^A\[\               ; Branch if string does not begin
                   4C   12  058F          BNEQ    30$                       ;  with bracket
     5D 8F    FF A544   91  0591          CMPB    -1(R5)[R4],#^A\]\         ; Branch if string does not end
                   44   12  0597          BNEQ    30$                       ;  with bracket
              54   02   C2  0599          SUBL2   #2,R4                     ; Discard brackets
                   55   D6  059C          INCL    R5                        ;
        65    54   2C   3A  059E          LOCC    #^A\,\,R4,(R5)            ; Locate group-member delimiter
                   39   13  05A2          BEQL    30$                       ; Branch on failure
        54    51   55   C3  05A4          SUBL3   R5,R1,R4                  ; <R4,R5> => group string
                   50   D7  05A8          DECL    R0                        ; <R0,R1> => member string
                   51   D6  05AA          INCL    R1                        ;
                   7E   D4  05AC          CLRL    -(SP)                     ; Allocate space from stack
                   5E   DD  05AE          PUSHL   SP                        ; Address of result
                   51   DD  05B0          PUSHL   R1                        ; Address of input string
                   50   DD  05B2          PUSHL   R0                        ; Size of input string
    00000000'GF    03   FB  05B4          CALLS   #3,G^FIL$CVT_OTB          ; Convert octal string to binary
              1D   50   E9  05BB          BLBC    R0,20$                    ; Branch on failure
        0C    A6   6E   B0  05BE          MOVW    (SP),XAB$W_MBM(R6)        ; Update member UIC value in XAB
                   5E   DD  05C2          PUSHL   SP                        ; Address of result
                   55   DD  05C4          PUSHL   R5                        ; Address of input string
                   54   DD  05C6          PUSHL   R4                        ; Size of input string
    00000000'GF    03   FB  05C8          CALLS   #3,G^FIL$CVT_OTB          ; Convert octal string to binary
              06   50   E9  05CF          BLBC    R0,10$                    ; Branch on failure
        0E    A6   6E   B0  05D2          MOVW    (SP),XAB$W_GRP(R6)        ; Update group UIC value in XAB
                   03   11  05D6          BRB     20$                       ; UIC has been successfully converted
              0C   A6   B4  05D8    10$:  CLRW    XAB$W_MBM(R6)             ; GRP is invalid, so also discard MBM
                   8E   D4  05DB    20$:  CLRL    (SP)+                     ; Deallocate space from stack
                        05DD
                        05DD    ;
                        05DD    ; Process the DAP PROSYS, PROOWN, PROGRP, and PROWLD fields.
                        05DD    ;
                        05DD
  50   04   00    50   A7   F0  05DD    30$:  INSV    DAP$W_PROSYS(R7),#0,#4,R0 ; Map system bits
  50   04   04    52   A7   F0  05E3          INSV    DAP$W_PROOWN(R7),#4,#4,R0 ; Map owner bits
  50   04   08    54   A7   F0  05E9          INSV    DAP$W_PROGRP(R7),#8,#4,R0 ; Map group bits
  50   04   0C    56   A7   F0  05EF          INSV    DAP$W_PROWLD(R7),#12,#4,R0; Map world bits
              08   A6   50   B0  05F5          MOVW    R0,XAB$W_PRO(R6)          ; Update protection mask in XAB
                        05F9    40$:  $CLRBIT #DAP$K_PRO_MSG,NWA$L_MSG_MASK(R7); Check it off from list
                        05  05FF          RSB                               ; Process next DAP message
```

```
                              0600          .SBTTL  NT$DECODE_NAM - UPDATE RESULTANT NAME
                              0600
                              0600  ;++
                              0600  ; Process (resultant) Name message from partner.
                              0600  ;--
                              0600
                              0600  NT$DECODE_NAM::                                 ; Entry point
                   00    E1   0600          BBC     #DAP$V_FILSPEC,-                ; Ignore NAM message if it does not
              13 40 A7        0602                  DAP$B_NAMETYPE(R7),10$          ;  contain a full filespec string
           50    44 A7   7D   0605          MOVQ    DAP$Q_NAMESPEC(R7),R0           ; Get descriptor of resultant name
         0170 CA    50   D0   0609          MOVL    R0,FWA$Q_QUOTED(R10)            ; Store it in quoted string buffer
   0174 DA    61    50   28   060E          MOVC3   R0,(R1),@FWA$Q_QUOTED+4(R10)
                              0614          $SETBIT #FWA$V_REMRESULT,(R10)          ; Flag receipt of resultant name string
                         05   0618  10$::   RSB                                     ; Exit
                              0619
                              0619          .END                                    ; End of module
```

```
$$.PSECT_EP                      = 00000000                 DAP$K_ANY                        = 00000000
$$COUNT                          = 00000002                 DAP$K_ATT_MSG                    = 00000002
$$RMSTEST                        = 0000001A                 DAP$K_BLN                          000000C0
$$RMS_PBUGCHK                    = 00000010                 DAP$K_CLOSE                      = 00000001
$$RMS_TBUGCHK                    = 00000008                 DAP$K_CMP_MSG                    = 00000007
$$RMS_UMODE                      = 00000004                 DAP$K_CYL                        = 00000001
BRO_RECV_CMP                       000001D4 R      01       DAP$K_DATATYP_D                  = 00000002
BRO_SEND_CMP                       000001BE R      01       DAP$K_FIX                        = 00000001
BUILD_MASK                         00000034 R      01       DAP$K_IDX                        = 00000020
DAP$B_ACCFUNC                      00000040                 DAP$K_KEY_MSG                    = 0000000A
DAP$B_ACCOPT                       00000041                 DAP$K_LBN                        = 00000002
DAP$B_AID                          00000050                 DAP$K_NAM_MSG                    = 0000000F
DAP$B_ALN                          00000044                 DAP$K_OPEN                       = 00000001
DAP$B_AOP                          00000045                 DAP$K_ORG_D                      = 00000000
DAP$B_BITCNT                       00000035                 DAP$K_PRO_MSG                    = 0000000E
DAP$B_BKS                          00000050                 DAP$K_RAT_D                      = 00000010
DAP$B_BKZ                          00000051                 DAP$K_REL                        = 00000010
DAP$B_BSZ                          00000052                 DAP$K_RFM_D                      = 00000001
DAP$B_CMPFUNC                      00000040                 DAP$K_SEQ                        = 00000000
DAP$B_DAN                          00000070                 DAP$K_STG                        = 00000000
DAP$B_DATATYPE                     00000044                 DAP$K_STM                        = 00000004
DAP$B_DBS                          0000007C                 DAP$K_STMCR                      = 00000006
DAP$B_DCODE_FID                    00000019                 DAP$K_STMLF                      = 00000005
DAP$B_DCODE_MAC                    0000001B                 DAP$K_SUM_MSG                    = 0000000C
DAP$B_DCODE_MSG                    0000001A                 DAP$K_TIM_MSG                    = 0000000D
DAP$B_DTP                          00000071                 DAP$K_UDF                        = 00000000
DAP$B_FAC                          00000042                 DAP$K_VAR                        = 00000002
DAP$B_FLAGS                        00000031                 DAP$K_VBN                        = 00000003
DAP$B_FLG                          00000048                 DAP$K_VFC                        = 00000003
DAP$B_FSZ                          00000051                 DAP$L_ALQ1                         0000004C
DAP$B_IAN                          0000006E                 DAP$L_ALQ2                         0000004C
DAP$B_IBS                          0000007D                 DAP$L_ATTMENU                      00000040
DAP$B_LAN                          0000006F                 DAP$L_CMWA                         00000030
DAP$B_LEN256                       00000034                 DAP$L_CRC_RSLT                     00000020
DAP$B_LENGTH                       00000033                 DAP$L_DCODE_STS                    00000018
DAP$B_LVL                          0000007E                 DAP$L_DEV                          00000068
DAP$B_NAMETYPE                     00000040                 DAP$L_DVB                          00000078
DAP$B_NOA                          00000045                 DAP$L_EBK                          00000078
DAP$B_NOK                          00000044                 DAP$L_FOP1                         00000064
DAP$B_NOR                          00000046                 DAP$L_FOP2                         00000044
DAP$B_NSG                          00000049                 DAP$L_HBK                          00000074
DAP$B_NUL                          0000006D                 DAP$L_KEYMENU                      00000040
DAP$B_ORG                          00000045                 DAP$L_LOC                          00000048
DAP$B_RAT                          00000047                 DAP$L_MRN                          00000058
DAP$B_REF                          0000006C                 DAP$L_MSG_MASK                     0000001C
DAP$B_RFM                          00000046                 DAP$L_RVB                          00000074
DAP$B_SHR                          00000043                 DAP$L_SBN                          0000007C
DAP$B_SIZ                          0000005C                 DAP$L_SSPWA                        00000080
DAP$B_SIZ_TMP                      0000004A                 DAP$L_SSP_CAP                      00000088
DAP$B_STREAMID                     00000032                 DAP$L_SSP_FLG                      00000084
DAP$B_TKS                          0000007F                 DAP$L_TEMP                         00000090
DAP$B_TYPE                         00000030                 DAP$M_ASCII                      = 00000001
DAP$B_X_FIELD                      00000024                 DAP$M_BITCNT                     = 00000008
DAP$C_BLN                          000000C0                 DAP$M_CMPFMT                     = 00000008
DAP$K_ACC_MSG                    = 00000003                 DAP$M_CR                         = 00000002
DAP$K_ACK_MSG                    = 00000006                 DAP$M_DATATYPE                   = 00000001
DAP$K_ALL_MSG                    = 0000000B                 DAP$M_DFTSPEC                    = 00000010
```

```
DAP$M_DMO                       = 00002000          DAP$V_DEQ2                      = 00000008
DAP$M_DSP_3NAM                  = 00000200          DAP$V_DSP_ALL                   = 00000002
DAP$M_DSP_ATT                   = 00000001          DAP$V_DSP_KEY                   = 00000001
DAP$M_EMBEDDED                  = 00000010          DAP$V_DSP_NAM                   = 00000008
DAP$M_FOP1                      = 00001000          DAP$V_DSP_PRO                   = 00000005
DAP$M_FTN                       = 00000001          DAP$V_DSP_SUM                   = 00000003
DAP$M_GET                       = 00000002          DAP$V_DSP_TIM                   = 00000004
DAP$M_GO_NOGO                   = 00000010          DAP$V_DUP                       = 00000000
DAP$M_IMAGE                     = 00000002          DAP$V_EMBEDDED                  = 00000004
DAP$M_LOAD                      = 00000001          DAP$V_FCS                       = 00000031
DAP$M_LOADIM                    = 00000001          DAP$V_FILSPEC                   = 00000000
DAP$M_LSA                       = 00000040          DAP$V_FSZ                       = 00000008
DAP$M_MACY11                    = 00000080          DAP$V_FTN                       = 00000000
DAP$M_MRS                       = 00000020          DAP$V_GEQ_V56                   = 00000024
DAP$M_MSE                       = 00000010          DAP$V_HRD                       = 00000000
DAP$M_NONFATAL                  = 00000001          DAP$V_NUL_CHR                   = 00000002
DAP$M_ORG                       = 00000002          DAP$V_ONC                       = 00000003
DAP$M_PRN                       = 00000004          DAP$V_PRN                       = 00000002
DAP$M_RAT                       = 00000008          DAP$V_RCK                       = 0000000F
DAP$M_RET_CRC                   = 00000008          DAP$V_STM_ONLY                  = 00000032
DAP$M_RFM                       = 00000004          DAP$V_VAXELAN                   = 00000035
DAP$M_SEGMENT                   = 00000040          DAP$V_VAXVMS                    = 00000034
DAP$M_SSP_FLG                   = 00000002          DAP$V_WCK                       = 0000000E
DAP$M_SYSPEC                    = 00000020          DAP$W_ALLMENU                     00000040
DAP$M_TMP1$                     = 00000020          DAP$W_BLS                         00000048
DAP$M_TMP2$                     = 000000C0          DAP$W_CHECK                       00000042
DAP$M_TMP3$                     = 00020000          DAP$W_DEQ1                        00000054
DAP$M_TMP4$                     = 01000000          DAP$W_DEQ2                        00000052
DAP$M_TMP5$                     = F0000000          DAP$W_DFL                         00000044
DAP$M_ZERO                      = 00000080          DAP$W_DISPLAY1                    0000004C
DAP$Q_ADT                         00000070          DAP$W_FFB                         00000072
DAP$Q_BDT                         00000060          DAP$W_IFL                         00000046
DAP$Q_CDT                         00000048          DAP$W_LRL                         00000070
DAP$Q_DCODE_FLG                   00000000          DAP$W_MRL                         00000072
DAP$Q_EDT                         00000058          DAP$W_MRS                         0000004A
DAP$Q_FILESPEC                    00000044          DAP$W_PARTNER                     00000006
DAP$Q_KNM                         00000064          DAP$W_POS                         0000004C
DAP$Q_MSG_BUF1                    00000008          DAP$W_POS_TMP                     0000004A
DAP$Q_MSG_BUF2                    00000010          DAP$W_PROGRP                      00000054
DAP$Q_NAMESPEC                    00000044          DAP$W_PROMENU                     00000040
DAP$Q_OWNER                       00000048          DAP$W_PROOWN                      00000052
DAP$Q_PASSWORD                    00000050          DAP$W_PROSYS                      00000050
DAP$Q_PDT                         00000068          DAP$W_PROWLD                      00000056
DAP$Q_RDT                         00000050          DAP$W_PVN                         00000042
DAP$Q_RUNSYS                      0000005C          DAP$W_RVN                         00000042
DAP$Q_SYSPEC                      00000038          DAP$W_SSP_MENU                    00000080
DAP$V_ALQ2                      = 00000005          DAP$W_SUMENU                      00000040
DAP$V_AOP                       = 00000002          DAP$W_TIMENU                      00000040
DAP$V_BKZ                       = 00000007          DAP$W_VERSION                     00000004
DAP$V_BLK                       = 00000003          DAP$W_VOL                         00000042
DAP$V_CBT                       = 00000017          DECODE_NXT_ALL                    0000026E R       01
DAP$V_CBT2                      = 00000002          DECODE_NXT_KEY                    00000257 R       01
DAP$V_CHG                       = 00000001          DONE                              00000285 R       01
DAP$V_CR                        = 00000001          FAB$B_FAC                       = 00000016
DAP$V_CTG                       = 00000007          FAB$B_FSZ                       = 0000003F
DAP$V_CTG2                      = 00000001          FAB$B_ORG                       = 0000001D
DAP$V_DEQ1                      = 0000000B          FAB$B_RAT                       = 0000001E
```

```
                                                 I  3
NTOOPEN                        NETWORK OPEN FILE               16-SEP-1984 00:03:45  VAX/VMS Macro V04-00        Page  29
Symbol table                                                    5-SEP-1984 16:20:58  [RMS.SRC]NTOOPEN.MAR;1            (18)

FAB$B_RFM                 = 0000001F                  IFB$W_DEQ                 = 00000062
FAB$C_FIX                 = 00000001                  IFB$W_LRL                 = 00000052
FAB$C_IDX                 = 00000020                  IFB$W_MRS                 = 00000060
FAB$C_REL                 = 00000010                  LOOP                        00000225 R      01
FAB$C_SEQ                 = 00000000                  LOOP1                       0000022D R      01
FAB$C_STM                 = 00000004                  LOOP2                       00000245 R      01
FAB$C_STMCR               = 00000006                  NT$BUILD_HEAD               ********    X   01
FAB$C_STMLF               = 00000005                  NT$BUILD_TAIL               ********    X   01
FAB$C_UDF                 = 00000000                  NT$CRC_INIT                 ********    X   01
FAB$C_VAR                 = 00000002                  NT$CVT_BN4_EXT              ********    X   01
FAB$C_VFC                 = 00000003                  NT$DECODE_ALL               0000043B RG     01
FAB$L_FOP                 = 00000004                  NT$DECODE_ALL_A             000004DE RG     01
FAB$L_MRN                 = 00000038                  NT$DECODE_KEY               0000037D RG     01
FAB$M_CBT                 = 00200000                  NT$DECODE_NAM               00000600 RG     01
FAB$M_CTG                 = 00100000                  NT$DECODE_PRO               0000057D RG     01
FAB$M_KFO                 = 40000000                  NT$DECODE_SUM               00000522 RG     01
FAB$M_RCK                 = 00800000                  NT$DECODE_TIM               0000053F RG     01
FAB$M_UFO                 = 00020000                  NT$EXCH_CNF                 ********    X   01
FAB$M_WCK                 = 00000200                  NT$GET_FAC_SHR              ********    X   01
FAB$V_BIO                 = 00000005                  NT$GET_FILESPEC             ********    X   01
FAB$V_BLK                 = 00000003                  NT$LCL_FOP                  ********    X   01
FAB$V_CBT                 = 00000015                  NT$MAP_DEV_CHAR             ********    X   01
FAB$V_CR                  = 00000001                  NT$MAP_FOP                  ********    X   01
FAB$V_CTG                 = 00000014                  NT$MOD_RAT                  00000366 RG     01
FAB$V_FTN                 = 00000000                  NT$OPEN                     00000000 RG     01
FAB$V_KFO                 = 0000001E                  NT$RECEIVE                  ********    X   01
FAB$V_NFS                 = 00000010                  NT$RECV_EXT_ATT             000001F1 RG     01
FAB$V_PRN                 = 00000002                  NT$SCAN_NAMBLK              ********    X   01
FAB$V_RCK                 = 00000017                  NT$SCAN_XABCHN              ********    X   01
FAB$V_UFO                 = 00000011                  NT$TRANSMIT                 ********    X   01
FAB$V_WCK                 = 00000009                  NT$UPDATE_FHC               00000315 RG     01
FAB$W_BLS                 = 0000003C                  NWA$B_ALLXABCNT             0000011C
FAB$W_DEQ                 = 00000014                  NWA$B_DAP_RAC               000000C9
FAC_BRO                     000001AF R      01        NWA$B_FILESYS               000000C5
FAIL                        00000288 R      01        NWA$B_KEYXABCNT             0000011D
FAIL1                       00000033 R      01        NWA$B_NETSTRSIZ             0000016F
FAIL2                       0000013D R      01        NWA$B_NODBUFSIZ             00000168
FAIL3                       000001AE R      01        NWA$B_ORG                   000000C6
FAIL_FOP                    0000013E R      01        NWA$B_OSTYPE                000000C4
FIL$CVT_OTB                 ********    X   01        NWA$B_RFM                   000000C7
FWA$Q_QUOTED              = 00000170                  NWA$B_RMS_RAC               000000C8
FWA$V_REMRESULT           = 00000035                  NWA$C_BLN                   00000800
IFB$B_BKS                 = 0000005E                  NWA$K_BLN                   00000800
IFB$B_FAC                 = 00000022                  NWA$K_IDX                 = 00000020
IFB$B_FSZ                 = 0000005F                  NWA$K_SEQ                 = 00000000
IFB$B_MODE                = 0000000A                  NWA$L_ALLXABADR             00000100
IFB$B_ORGCASE             = 00000023                  NWA$L_DATXABADR             00000104
IFB$B_RAT                 = 00000051                  NWA$L_DEV                   000000C0
IFB$B_RFMORG              = 00000050                  NWA$L_FHCXABADR             00000108
IFB$L_DEVBUFSIZ           = 00000048                  NWA$L_KEYXABADR             0000010C
IFB$L_HBK                 = 00000070                  NWA$L_MSG_MASK              000000D4
IFB$L_NWA_PTR             = 0000003C                  NWA$L_PROXABADR             00000110
IFB$S_ORG                 = 00000004                  NWA$L_RDTXABADR             00000114
IFB$V_BIO                 = 00000005                  NWA$L_SAVE_FLGS             00000128
IFB$V_BRO                 = 00000006                  NWA$L_SUMXABADR             00000118
IFB$V_DAP_OPEN            = 0000003D                  NWA$L_THREAD                000000FC
IFB$V_ORG                 = 00000004                  NWA$L_XLTATTR               00000238
```

```
                                                        J  3
NTOOPEN                          NETWORK OPEN FILE                 16-SEP-1984 00:03:45  VAX/VMS Macro V04-00        Page  30
Symbol table                                                        5-SEP-1984 16:20:58  [RMS.SRC]NTOOPEN.MAR;1           (18)

NWA$L_XLTBUFFLG                  0000022C                XAB$B_AID                  = 00000017
NWA$L_XLTCNT                     00000228                XAB$B_ALN                  = 00000009
NWA$L_XLTMAXINDX                 00000234                XAB$B_AOP                  = 00000008
NWA$L_XLTSIZ                     00000230                XAB$B_ATR                  = 00000009
NWA$Q_ACS                        00000244                XAB$B_BKZ                  = 00000016
NWA$Q_BIGBUF                     00000170                XAB$B_BLN                  = 00000001
NWA$Q_BLD                        000000F0                XAB$B_DAN                  = 0000000A
NWA$Q_FLG                        00000000                XAB$B_DBS                  = 0000000D
NWA$Q_INODE                      0000025C                XAB$B_DTP                  = 00000013
NWA$Q_IOSB                       000000D8                XAB$B_FLG                  = 00000012
NWA$Q_LNODE                      00000160                XAB$B_HSZ                  = 00000017
NWA$Q_LOGNAME                    0000023C                XAB$B_IAN                  = 00000008
NWA$Q_NCB                        00000264                XAB$B_IBS                  = 0000000C
NWA$Q_RCV                        000000E0                XAB$B_LAN                  = 00000009
NWA$Q_SAVE_DESC                  00000120                XAB$B_LVL                  = 0000000B
NWA$Q_XLTBUF1                    0000024C                XAB$B_NOA                  = 00000008
NWA$Q_XLTBUF2                    00000254                XAB$B_NOK                  = 00000009
NWA$Q_XMT                        000000E8                XAB$B_NSG                  = 00000014
NWA$T_ACSBUF                     0000026C                XAB$B_NUL                  = 00000015
NWA$T_AUXBUF                     000005E0                XAB$B_REF                  = 00000017
NWA$T_DAP                        00000000                XAB$B_RFO                  = 00000008
NWA$T_INODEBUF                   000004AC                XAB$B_SIZ                  = 0000002E
NWA$T_ITM_ATTR                   00000200                XAB$B_TKS                  = 00000016
NWA$T_ITM_END                    00000224                XAB$C_CYL                  = 00000001
NWA$T_ITM_LST                    00000200                XAB$C_DATLEN               = 0000002C
NWA$T_ITM_MAXINDX                00000218                XAB$C_LBN                  = 00000002
NWA$T_ITM_STRING                 0000020C                XAB$C_VBN                  = 00000003
NWA$T_NCBBUF                     0000052C                XAB$L_ALQ                  = 00000010
NWA$T_NODEBUF                    00000169                XAB$L_DVB                  = 0000003C
NWA$T_RCVBUF                     000001A0                XAB$L_EBK                  = 00000010
NWA$T_SCAN                       00000100                XAB$L_HBK                  = 0000000C
NWA$T_TEMP                       00000120                XAB$L_KNM                  = 00000038
NWA$T_XLTBUF1                    000002AC                XAB$L_LOC                  = 0000000C
NWA$T_XLTBUF2                    000003AC                XAB$L_NXT                  = 00000004
NWA$T_XMTBUF                     000003C0                XAB$L_RVB                  = 0000000E
NWA$V_LAST_MSG                 = 00000000                XAB$L_SBN                  = 00000028
NWA$V_NODECODE                 = 00000002                XAB$L_UIC                  = 0000000C
NWA$W_BUILD                      000000D2                XAB$Q_BDT                  = 00000024
NWA$W_DAPBUFSIZ                  000000CA                XAB$Q_CDT                  = 00000014
NWA$W_DIR_OFF                    000000CC                XAB$Q_EDT                  = 0000001C
NWA$W_DISPLAY                    000000D0                XAB$Q_RDT                  = 0000000C
NWA$W_FIL_OFF                    000000CE                XAB$V_CBT                  = 00000005
NWA$W_JNLXABJOP                  0000011E                XAB$V_CHG                  = 00000001
OPEN_UPDATE_FAB                  00000289 R       01     XAB$V_CTG                  = 00000007
PART1                            00000050 R       01     XAB$V_DUP                  = 00000000
PART2A                           0000007E R       01     XAB$V_HRD                  = 00000000
PART2B                           0000009D R       01     XAB$V_NUL                  = 00000002
PART3                            000000CB R       01     XAB$V_ONC                  = 00000001
PIO$A_TRACE                      ********    X    01     XAB$W_DEQ                  = 00000014
RECV_ACK                         00000197 R       01     XAB$W_DFL                  = 0000001C
RECV_ATT                         00000141 R       01     XAB$W_DXQ                  = 0000001A
RECV_EXT_ATT                     0000017E R       01     XAB$W_FFB                  = 00000014
RECV_NAM                         00000183 R       01     XAB$W_GRP                  = 0000000E
SEND_ACC                         000000ED R       01     XAB$W_IFL                  = 0000001A
SEND_ATT                         0000004A R       01     XAB$W_LRL                  = 0000000A
SUC                              000001A7 R       01     XAB$W_MBM                  = 0000000C
TPT$L_NTOPEN                     ********    X    01     XAB$W_MRL                  = 00000018
```

```
XAB$W_MRZ                        = 00000018
XAB$W_POS                        = 0000001E
XAB$W_PRO                        = 00000008
XAB$W_PVN                        = 0000000A
XAB$W_RVN                        = 00000008
XAB$W_VOL                        = 0000000A

                                             +-----------------+
                                             ! Psect synopsis !
                                             +-----------------+

PSECT name                       Allocation          PSECT No.  Attributes
----------                       ----------          ---------  ----------
.  ABS  .                        00000000  (     0.)  00 (  0.)  NOPIC    USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
NF$NETWORK                       00000619  (  1561.)  01 (  1.)    PIC    USR   CON   REL   GBL NOSHR   EXE   RD   NOWRT NOVEC BYTE
$ABS$                            00000800  (  2048.)  02 (  2.)  NOPIC    USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE

                                          +------------------------+
                                          ! Performance indicators !
                                          +------------------------+

Phase                       Page faults    CPU Time       Elapsed Time
-----                       -----------    --------       ------------
Initialization                       29    00:00:00.10    00:00:01.08
Command processing                  106    00:00:00.56    00:00:03.51
Pass 1                              479    00:00:21.36    00:00:56.48
Symbol table sort                     0    00:00:02.56    00:00:05.86
Pass 2                              207    00:00:04.44    00:00:11.66
Symbol table output                  54    00:00:00.40    00:00:02.65
Psect synopsis output                 1    00:00:00.02    00:00:00.02
Cross-reference output                0    00:00:00.00    00:00:00.00
Assembler run totals                878    00:00:29.46    00:01:21.27

The working set limit was 1650 pages.
112810 bytes (221 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1711 non-local and 88 local symbols.
1113 source lines were read in Pass 1, producing 18 object records in Pass 2.
41 pages of virtual memory were used to define 40 macros.

                                          +----------------------------+
                                          ! Macro library statistics !
                                          +----------------------------+

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[RMS.OBJ]RMS.MLB;1                       32
_$255$DUA28:[SYSLIB]STARLET.MLB;2                     4
TOTALS (all libraries)                               36

2157 GETS were required to define 36 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:NTOOPEN/OBJ=OBJ$:NTOOPEN MSRC$:NTOOPEN/UPDATE=(ENH$:NTOOPEN)+LIB$:RMS/LIB
```

0317 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

NTØSCNXAB
LIS

RMØCACHE
LIS

NTØRENAME
LIS

NTØOPEN
LIS

RMØBUFMGR
LIS

NTØSEARCH
LIS

NTØPUT
LIS

RMØACCESS
LIS